IN UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

VERTICAL COMPUTER SYSTEMS, INC.,	)	Civil Action No. 2:07-cv-00144 DF-CE
	)	
Plaintiff,	)	Honorable Judge David Folsom
	)	
v.	)	
	)	Magistrate Judge Charles Everingham
MICROSOFT CORPORATION,	)	
	)	
Defendant.	)	

VERTICAL'S OPENING BRIEF SUPORTING
ITS CLAIM CONSTRUCTIONS FOR THE '744 PATENT

T. John Ward, Jr.
WARD & SMITH LAW FIRM
111 West Tyler Street
Longview, Texas 75601
(903) 757-6400
Fax: (903) 747-2323

Eric M. Albritton
ALBRITTON LAW FIRM
P.O. Box 2649
Longview, Texas 75606
(903) 757-8449
Fax: (903) 758-7397

Raymond P. Niro
Vasilios D. Dossas
Sally Wiggins
NIRO, SCAVONE, HALLER & NIRO
181 West Madison Street, Suite 4600
Chicago, Illinois 60602
(312) 236-0733
Fax: (312) 236-3137

Attorneys for Plaintiff

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

FEDERAL CASES

Page(s)

I. INTRODUCTION

Plaintiff Vertical Computer Systems, Inc. ("Vertical") submits its claim constructions for U.S. Patent No. 6,826,744 titled "System and Method for Generating Web Sites in an Arbitrary Object Framework" ("the '744 patent," attached as Exhibit A). The parties have filed a joint claim construction chart consistent with this Court's Docket Control Order in which the parties identified the disputed terms. (Vertical also attaches a copy of the file history of the '744 patent as Exhibit B.)

The key distinction between the parties' claims constructions is founded upon Microsoft's attempts to import limitations from the preferred embodiments and from specific examples provided in the specification of the '744 patent into the claims. Microsoft ignores the fact that the patent uses words of inclusion rather than words or expressions of manifest exclusion or restriction. The specification provides many examples for the term "arbitrary objects," for example, but nowhere does it suggest that the term "arbitrary objects" in the claims is limited to one of these examples. The '744 patent specification uses broad exemplary language, not limiting language and the claims must be interpreted consistently with the specification. It is important to identify boundaries and not merely descriptions of the included scope, as Microsoft has done.

II. BACKGROUND OF THE INVENTION

The technical field of the '744 patent is complex computer programming. Groups of software developers working together often create such programs, and different groups of people develop different pieces of the program. These "pieces" are normally called "objects" in classical programming. A master developer then puts the pieces together, like a puzzle, to form the complete computer program. The invention of the '744 patent is a major advance in how such a complex computer program is built. At the heart of the

invention is the new concept of "arbitrary objects" disclosed in the '744 patent. The "arbitrary objects" allow various groups of developers to work independently yet connect their work products (or interface one with the other) quickly and effectively.

A. Classic Objects

Vertical will use the label "classic" for various objects (i.e., program pieces) to differentiate between the ordinary objects in existence prior to the invention of the '744 patent and the "arbitrary objects" described and claimed in the '744 patent. In traditional frameworks, a classic object could be a set of computer instructions grouped together to perform a function. Vertical provides the following specific example just to illustrate classical prior art objects:

A programmer in the field of accounting can create an object (a small functionality piece for a master accounting program) that will calculate the net payment for an employee. In this example, the object may have the name "CalculateNetPay", and the master program has to retrieve it with several parameters: social security of the employee, hourly pay, number of hours worked in the period, number of overtime hours worked in the period and number of hours taken as sick hours. The "CalculateNetPay" object will then add the number of sick hours to the number of worked hours and multiply the results by the hourly wage, and store that result temporarily in a memory field named "S1". It will then multiply the number of overtime hours by the hourly wage, then multiply this result by 1.25 (to add 25% of overtime pay), and add this result to S1. It will then collect all of the normal payroll deductions such as federal income tax, etc. and subtract them from S1. It will read the database with all the information about the employee by using the social security number, and print the pay stub with all of the employee information and the amounts to be paid (contained in S1).

Someone using this classic object must know in advance which parameters this object uses. This process of tying the basic or main program with the object's parameters is called "binding". A little mistake, such as mistaking the social security number for the hours worked, will cause this object to malfunction, and the entire application will malfunction, or as they say in computer circles, will "crash". In other words, a programmer of the classic object "CalculateNetPay" must know in advance that in using this object he or she must retrieve it in the following way: CalculateNetPay with parameters SocialSecurityNumber, HourlyPay, HoursWorked, OvertimeHours, SickHours. Classic objects by their nature are pre-determined, and the users of those objects need to know how to use them, which parameters to use. It is absolutely critical that the user supply these parameters in the correct order. A switch in order will cause the object to malfunction.

B. Arbitrary Objects of the '744 Patent

The '744 Patent creates a new way of interfacing program units -- using Arbitrary Objects. These objects can be retrieved by name only. There is no need for the user of an arbitrary object to know all the parameters and usage of the arbitrary object. The arbitrary object framework will take care of the binding necessary. If the "CalculateNetPay" Functional Object were arbitrary, all that the user would need to know to use this object is: CalculateNetPay. The need to pass parameters does not exist. The Arbitrary Object Framework will take care of binding. During the creation of the arbitrary functionality object named "CalculateNetPay," it's parameters and needs are stored in the arbitrary object framework's object library. When retrieving an arbitrary object, the user or programmer will only mention the name. The arbitrary object framework will find the object in the library, check what data it needs, and pass the required data to the arbitrary object.

In summary, the present invention substantially simplifies the interface between one program unit and another unit.

III. THE SPECIFICATION'S ROLE IN CLAIM CONSTRUCTION

The Court of Appeals for the Federal Circuit in Phillips v. AWH Corp., 415 F.3d 1303, 1314 (Fed. Cir. 2005) clarified that, when construing claims, the primary focus remains on the claims, both asserted and unasserted. Although the specification is the single best source for understanding the meaning of the claims, it cannot be used as a tool to redefine otherwise clearly understood phrases and terms. Phillips, 415 F.3d at 1323. "For instance, although the specification often describes very specific embodiments of the invention, [the Federal Circuit] has repeatedly warned against confining the claims to those embodiments." Id. "That is not just because section 112 of the Patent Act requires that the claims themselves set forth the limits of the patent grant, **but because persons of ordinary skill in the art rarely would confine their definitions of terms to the exact representations depicted in the embodiments**." Id. (Emphasis added.) "In sum, subject to any clear and unmistakable disavowal of claim scope, the term[s] . . . take the full breadth of [their] ordinary meaning . . . ." Innova/Pure Water, Inc. v. Safari Water Filtration Systems, Inc., 381 F.3d 1111, 1120 (Fed. Cir. 2004). "And, even where a patent describes only a single embodiment, claims will not be read restrictively unless the patentee has demonstrated a clear intention to limit the claim scope using words or expressions of manifest exclusion or restriction." Id. at 1117.

"Differences among claims can also be a useful guide in understanding the meaning of particular claims terms." Phillips, 415 F.3d at 1314. For example, the doctrine of claim differentiation creates a rebuttable presumption that each claim in a patent has different scope. Sunrace Roots Enter., Co. v. SRAM Corp., 336 F.3d 1298, 1302-1303

(Fed. Cir. 2003). That presumption, however, "is especially strong when the limitation in dispute is the only meaningful difference between an independent claim and dependent claim, and one party is urging that the limitation in the dependent claim should be read into the independent claim." Id. at 1303.

IV. THE CLAIMS OF THE '744 PATENT

The invention of the '744 patent is centered on using "arbitrary objects" in creating computer programs or applications:

> The method of this invention includes creating arbitrary objects, managing the arbitrary objects throughout their life cycle, and deploying the arbitrary objects in a design framework for use in complex computer applications. (Column 2, lines 14-18 of the '744 patent, Exhibit A.)

The claims of the '744 patent describe just that:

> 1. A method for generating a computer application on a host system in an **arbitrary object framework that separates a content of said computer application, a form of said computer application and a functionality of said computer application**, said method comprising:
>
> creating **arbitrary objects** with corresponding **arbitrary names** of various object types for generating said **content** of said computer application, said **form** of said computer application, and said **functionality** of said computer application;
>
> managing said **arbitrary objects** in an object library; and
>
> deploying said **arbitrary objects** from said object library into a design framework to create said computer application. (Emphasis added. Column 7, lines 16-29, Exhibit A.)

The bold portions are the terms presented for construction by the parties. Claim 1 is one of two asserted independent claims. The other is claim 26 which is similar to claim 1 but limits "computer application" to "web site." (The other eight asserted and dependent claims include 3-5, 17, 18, 21, 27 and 28.)

Claim 1 and claim 26 are method claims that have an extended preamble, the portion before the words "said method comprising," and three steps, including creating arbitrary objects, managing arbitrary objects and deploying arbitrary objects. The term "arbitrary object framework" and the portion of the preamble that states what an arbitrary object framework may accomplish ("that separates….") do not serve as antecedents for any term in the body of the claim (the three steps).

Generally, the preamble of a claim is non-limiting. DeGeorge v. Bernier, 768 F.2d 1318, 1322, n.3 (Fed Cir. 1985). As the Federal Circuit has explained,

> [i]f…the body of the claim fully and intrinsically sets forth the complete invention, including all of its limitations, and the preamble offers no distinct definition of any of the claimed invention's limitations, but rather merely states, for example, the purpose or intended use of the invention, then the preamble is of no significance to claim construction because it cannot be said to constitute or explain a claim limitation.

Pitney Bowes, Inc. v. Hewlett-Packard Co., 182 F.3d 1298, 1305 (Fed. Cir. 1999). A claim preamble is limiting only "if it recites essential structure or steps, or if it is 'necessary to give life, meaning, and vitality' to the claim. Catalina Marketing International v. Coolsavings.com, Inc., 289 F.3d 801, 808 (Fed. Cir. 2002).

Here, since the preamble primarily only states terms that are re-stated in the body of the claim, the Court should not consider anything in the preamble to be a claim limitation.

V. PROPOSED CONSTRUCTIONS

A. The Definition of "Objects"

Since the '744 invention is focused on "arbitrary objects," Vertical will begin with this term, which is the only term that Vertical believes requires construction. Vertical will then consider the remaining terms presented for construction.

The ‘744 patent does not use the term “object” apart from the word “arbitrary.” This makes sense, because the ‘744 patent introduces “arbitrary objects” into this technological field. The word “object” in both the classic context and in the context of the ‘744 invention simply means a programming piece or unit. The difference between Vertical’s definition of “object” and Microsoft’s definition is that in stating what that programming unit contains, Vertical lists the possible contents of a piece of computer program – form, function and content – in the disjunctive (A or B or C) while Microsoft lists them in the conjunctive (A and B and C) .

Vertical	Microsoft
An entity that can have form, content or functionality, or any combination of form, content and functionality.	This term does not require a construction separate from the construction of “arbitrary object.” To the extent that the Court decides to construe this term separately, Microsoft proposes that it mean “a combination of application logic and data.”

But, one of the goals of this “arbitrary object” invention, as stated in the preamble of each claim and throughout the file history of the ‘744 patent is separation of form, function and content – not the combination of these elements. Therefore, Microsoft’s definition of “object” simply cannot be the proper definition. Vertical submits that in the context of the disclosure of the ‘744 patent, “object” means a computer program piece [unit or entity] that may contain form, function, or content or a combination of these elements. (See Column 1, lines 11-12.) The entire disclosure of the ‘744 patent (Exhibit A) compels this definition.

B. <u>The Definition of “Arbitrary Objects”</u>

Turning now to the construction of “arbitrary objects,” the parties suggest the following definitions:

Vertical	Microsoft
An object that can be created independently by individual preference and that can be accessed by name.	Any combination of application logic and data desired by the developer that is interchangeable with another object of another type.

The part of the specification that provides a definition for "arbitrary object" is the following:

> Many functions are stored within an object library on an arbitrary object framework such that those functions can be accessed by name arbitrarily. This is in contrast to a traditional model where the function must be explicitly invoked with all its parameters included. (Column 5, lines 42-46, Exhibit A.)
>
> * * *
>
> A critical distinction between the present invention and previous object oriented development systems is the need to know how a function can be called and what to expect it to return, rather than just knowing the function's name. This means that typically the system administrator calls the name of an object and passes parameters to the object. Any and all variable information or environmental information can be available to every object. The environment space can be available to all objects executed and an object can arbitrarily take advantage of any of the environmental information, depending on the design of the object. (Emphasis added. Column 5, line 62 to column 6, line 5, Exhibit A.)

The '744 patent thus provides a clear distinction between the invention that it describes and claims and the prior art; and in the process provides a definition for "arbitrary objects." An arbitrary object is simply a program piece that can be retrieved by using only its name.

Microsoft disregards this critical intrinsic evidence and instead selectively collects self-serving specific examples and language to improperly import them into the claims. The specification includes a large number of examples, descriptions, and words of inclusion for "arbitrary objects:" Those examples include the following:

> These arbitrary objects may include encapsulated legacy data, legacy systems and custom programming logic from essentially any source in which they may reside. Any language supported by the host system, or any

> language that can be interfaced to by the host system, can be used to generate an object within the application. (Column 2, lines 29-34, Exhibit A.)
>
> * * *
>
> Arbitrary objects can include text file pointers, binary file pointers, compiled executables, scripts, data base queries, shell commands, remote procedure calls, global variables, and local variables. (Column 3, lines 43-46, Exhibit A.)
>
> * * *
>
> Arbitrary objects may include any combination of application logic and data desired by a developer. Arbitrary objects can include text file pointers, binary file pointers, compiled executable scripts, database queries, shell commands, remote call procedures, global variables and local variables. Arbitrary objects may also include cached data queries and executables. The arbitrary object framework allows arbitrary objects to be referenced in a consistent manner regardless of the type of object. Also, the arbitrary object framework allows local arbitrary objects to either override global parent arbitrary objects or inherit capabilities and data from the global parent arbitrary object.
>
> Arbitrary objects can execute any function that can be run or understood by the host computer system so that any underlying functionality of the operating system used by the host system can be defined as an object within the arbitrary framework. Legacy data, document objects, CPI programs, and database queries can all be encapsulated as objects within the arbitrary framework. The arbitrary object can be accessed by an arbitrary object name. Arbitrary objects are not tied to their functionality. One arbitrary object can be easily replaced with another arbitrary object of another type. (Emphasis added. Column 4, lines 21-41, Exhibit A.)
>
> * * *
>
> Arbitrary objects can be managed in an object library. The life cycle of the arbitrary objects may be managed in a 45 consistent manner using revision tracking, roll-back, and sign-off. (Column 4, lines 42-45, Exhibit A.)
>
> * * *
>
> Arbitrary objects can be deployed from the object library into a container page to generate the web site. (Column 4, lines 55-56, Exhibit A.)

Microsoft cherry picks only a few of these words of inclusion to create a definition that suits its purposes. The highlighted sections in the passages quoted above are those from which Microsoft composed its definition, but there does not exist any reason for choosing these words over the others in the preceding quotations. Accordingly, Microsoft's

proposed definition is utterly capricious. Microsoft evidently chose this definition to try to avoid the application of claim differentiation, as many of the different attributes of arbitrary objects appear in dependent claims such as claims 2-15, 22, 27-38, 44, and 48-49. Therefore, Microsoft cannot import them into the independent claims (1 and 26).

C. The Definition of "Arbitrary Name"

The parties have proposed the following definitions for "arbitrary name:"

Vertical	Microsoft
An identifier assigned to an arbitrary object.	Any name desired by the developer that is all that is needed to access the object.

The specification of the '744 patent (Exhibit A) provides the following discussions about "arbitrary name:"

> Many functions are stored within an object library on an arbitrary object framework such that those functions can be accessed by name arbitrarily. This is in contrast to a traditional model where the function must be explicitly invoked with all its parameters included. (Emphasis added. Column 5, lines 41-45, Exhibit A.)
>
> * * *
>
> All that is needed is the name of the function in order to access the function. (Emphasis added. Column 5, lines 53-54, Exhibit A).
>
> * * *
>
> A critical distinction between the present invention and previous object oriented development systems is the need to know how a function can be called and what to expect it to return, rather than just knowing the function's name. (Emphasis added. Column 5, lines 61-64, Exhibit A.)
>
> * * *
>
> The object name is essentially just another variable in the environment. (Column 6, lines 14-15, Exhibit A.)

Thus, as the specification makes abundantly clear, an "arbitrary name" is just the name of the arbitrary object.

Apparently realizing that it cannot simply discard the true definition of arbitrary objects, Microsoft injects the central feature of "arbitrary objects" into the definition for "arbitrary name," where it does not belong. Microsoft cannot burden the term "arbitrary name" with the definition of "arbitrary object" and provide a restrictive definition for "arbitrary objects" which it hopes will allow it to avoid infringement.

D. The Definition of "Content"

The parties propose the following definitions for the term "content:"

Vertical	Microsoft
Data.	Written, recorded, or illustrated documentation of the computer application [web site], such as photographs, illustrations, product marketing material, and news articles that can be created by writers, photographers, artists, reporters, or editors.

The specification lists a large number of examples for "content." But, all those lists are just that: examples. They do not define the word "content" because there are more examples that the patent does not list. Taken as a whole, the descriptions in Column 1, lines 18-22, column 5, lines 18-27, and column 6, lines 32-39 all point to one construction: data. In fact, in the last of these three citations (column 6, lines 32-39) the patent equates content and data.

> For instance, if a company would like to roll out a new look or syndicate its content and functionality to another business, this can be easily accomplished using the present invention. Since there is no application code resident in a web page itself, the same data can be repackaged in a number of different ways across multiple sites. [Emphasis added.]

The examiner of the '744 patent did just that when comparing the claims with the prior art in an office action dated April 3, 2003:

> Furthermore, see figure 10 on page 22, with its BLO to generate content (data), Presentation Objects to represent form, and its BPO to represent functionality. [Emphasis added.]

(See right-hand column on page 3 of the office action. Attached separately as Exhibit C) The examiner also equated content with data. Accordingly, Vertical respectfully asserts that the proper definition for "content" is data.

Once again, Microsoft adopts examples and words of inclusion and improperly uses them as boundaries, importing them into the claims. In column 1, lines 18-22 (Exhibit A), under the heading "BACKGROUND OF THE INVENTION," the specification states:

> Form includes informative content. Informative content can include written, recorded, or illustrated documentation, such as photographs, illustrations, product marketing material, and news articles. Content can be created by writers, photographers, artists, reporters, or editors.

This portion of the patent simply provides examples and not a definition. Microsoft conveniently uses the back end of the second (middle) sentence of this quote and leaves out the most important part – "can include." "Content" can include these examples; but the examples do not belong in the definition. (The patent repeats a similar description of examples for "content" in column 3, lines 23 to 38.)

E. The Definition of "Form"

As with content, the '744 patent specification includes a listing of examples for "form." Column 1, lines 14-16, and column 3, lines 28-39, list graphic designs, user interfaces, and graphical representations. The '744 patent also states that form "includes informative content" (column 1, line 17), which would indicate that form may include data, resulting in some combination of form and content. But, this listing of examples provided in the patent simply contains examples of formatting. Therefore, that is the definition that

is proper. Vertical proposes that definition, while Microsoft instead again tries to import the specific examples into the claims:

Vertical	Microsoft
Formatting.	The look of the computer application [web site], including graphic designs, user interfaces, and graphical presentations that can be created by a designer or group of designers.

Although part of Microsoft's definition, "the look of the computer application," is similar to Vertical's definition, Microsoft's listing of specific examples (i.e., the importation of those examples into the claims) is completely improper, as is the insertion of "that can be created by a designer or group of designers." This last phrase is a modifier meant to provide an example of who can create form. But machines may also create graphic designs, etc. Thus, Microsoft's definition is not correct.

F. The Definition of "Functionality"

In column 1, lines 16-17, the '744 patent (Exhibit A) states "[f]unction includes logical functionality which can be software code created by a programmer or group of programmers." In column 3, lines 30-31, the '744 patent provides that "[f]unction 14 can include the logical functionality of software code and scripts." The parties proposed the following definitions:

Vertical	Microsoft
Software code.	Software code or scripts to supplement the actions of the computer application [web site] that can be created by a programmer or group of programmers.

Software code is a broad term which includes scripts. Again, part of Microsoft's definition is similar to Vertical's definition, but the part that states: "that can be created by a programmer or group of programmers" is objectionable. Microsoft cannot import a permissive phrase from the specification and transform it into a restrictive term in their definition. There is no clear statement of exclusion for machine-generated functions; and thus, Microsoft's definition is improper.

G. The Definition "Arbitrary Object Framework"

The parties have proposed the following constructions for "arbitrary object framework:"

Vertical	Microsoft
A hierarchical system that can separate content, form and functionality to generate a product, and facilitates creation of arbitrary objects, management of arbitrary objects and deployment of arbitrary objects.	This term is indefinite. To the extent that the term can be given any meaning, Microsoft proposes that it means: a framework that allows arbitrary objects to be referenced in a consistent manner regardless of type.

"Arbitrary object framework" does not appear in the body of claims 1 and 26, but in the preamble of those claims. The only term that appears in the body of claim 1 which includes the word "framework" is "design framework," and, as claim 26 makes clear, an example of "design framework" is a container page. Thus, Vertical submits that the Court may either adopt the definition that Vertical has proposed for "arbitrary object framework" for the reasons stated below or dismiss it altogether because "arbitrary object framework" is not a limitation of the claims.

Generally, the preamble does not limit a claim as previously discussed. For example, where a preamble term serves as a convenient label for the invention, it does not limit the claim scope. Storage Tech. Corp. v. Cisco Systems Inc., 329 F.3d 823 (Fed. Cir. 2003). Also, in Intirtool, Ltd. v. Texas Corp., 369 F.3d 1289 (Fed. Cir. 2004), the Court of Appeals for the Federal Circuit considered whether the preamble of a claim-in-suit is a limitation of the invention defined by that claim. It provided the following guidelines:

> In general, a claim preamble is limiting if "it recites essential structure or steps, or if it is necessary to give 'life, meaning, and vitality' to the claim" *Id. at 808* (quoting *Pitney Bowes, Inc. v. Hewlett-Packard Co., 182 F.3d 1298, 1305 (Fed. Cir. 1999))*. However, if the body of the claim "describes a structurally complete invention such that deletion of the preamble phrase does not affect the structure or steps of the claimed invention," *id. at 809*, the preamble is generally not limiting unless there is "clear reliance on the preamble during prosecution to distinguish the claimed invention from the prior art," *id. at 808*.
>
> * * *
>
> Nor do we find in the prosecution history clear reliance specifically on the preamble, rather than on the structural limitations set forth in the body of the claim, which if present might provide a basis for transforming the preamble into a claim limitation. *See id.* ("Clear reliance on the preamble into a claim limitation.").
>
> * * *
>
> In context, the references to "punching and connecting" in the prosecution history appear to simply recite "benefits or features" of the claimed invention, and we see no "clear reliance on those benefits or features as patentably significant." *Catalina Mktg., 289 F.3d at 809*. In short, the preamble adds nothing to this highly detailed claim and thus cannot be considered to give "life, meaning, and vitality" to it. *See Kropa v. Robie, 38 C.C.P.A. 858, 187 F.2d 150, 152, 1951 Dec. Comm'r Pat. 177 (CCPA 1951)*. We hold that the preamble is not a limitation of claim 1.

369 F.3d at 1295-96.

The '744 patent does not indicate a reliance on both the preamble and claim body to define the claimed invention; and the preamble is not essential to understand the

limitations or terms in the claim body. See, Eaton Corp. v. Rockwell Int'l Corp., 323 F.3d 1332 (Fed. Cir. 2003). The bodies of claims 1 and 26 do not need "arbitrary object framework" to act as an antecedent for any term they contain. Thus, this term is not a limitation, and the Court need not construe it.

Should the Court decide that "arbitrary object framework" is a limitation, then Vertical proposes its definition as the only possible construction. Microsoft first declares that the term "arbitrary object framework" is indefinite even though there cannot be any doubt what "arbitrary object" means and even though the claims themselves describe the "arbitrary object framework" immediately after its appearance in the claims. For example, claim 1 states "arbitrary object framework that separates a content of said computer application, a form of said computer application and a functionality of said computer application,…." Thus, the claim language itself defines this term. See, Phillips v. AWH Corp., 415 F.3d 1303, 1312 (Fed. Cir. 2005)

> ("[W]e look to the words of the claims themselves to define the scope of the patented invention…. [T]he Supreme Court made clear that the claims are of primary importance, in the effort to ascertain precisely what it is that is patented.") (Citations omitted).

Given this modifier after the term "arbitrary object framework" and the body of the claims, Vertical's definition is the proper one.

Furthermore, the specification provides that:

> The present invention provides a system and method for using a hierarchical, arbitrary object framework for generating software applications. The method separates content, form, and function of the software application so that each can be accessed or modified independently. The method of this invention includes creating arbitrary objects, managing the arbitrary objects in an object library, and deploying the arbitrary objects in a design framework for use in computer applications. (Emphasis added. Column 3, lines 14-23, Exhibit A).

* * *

> The hierarchical framework separates content 10, form 12, and functionality 14 to generate product 16. (Emphasis added. Column 3, lines 32-34, Exhibit A.)

This quote confirms that an "arbitrary object framework" is a hierarchical system that can separate content, form and functionality and facilitates the specific steps of the method defined by the claims. Vertical submits that "arbitrary object framework" can only mean what Vertical has proposed.

Microsoft's proposal has a number of fundamental flaws that render it patently improper. First, Microsoft's definition contains the term that it defines. Second, Microsoft has done exactly what it did with respect to "arbitrary objects" – it took words of inclusion for "arbitrary objects" from the specification and turned them into words of exclusion for "arbitrary object framework." The specification provides:

> The arbitrary object framework allows arbitrary objects to be referenced in a consistent manner regardless of the type of object. Also, the arbitrary object framework allows local arbitrary objects to either override global parent arbitrary objects or inherit capabilities and data from the global parent arbitrary object. (Column 4, lines 26-32, Exhibit A.)

Microsoft chose the first sentence of this quote for its proposed definition. It did not choose the second sentence, which is just as viable in Microsoft's approach to construing the claims. One reason that Microsoft avoided the second sentence is that claim differentiation would render that selection even more improper. Claims 13 and 14 describe the features of the second sentence; thus, they could not be part of the dependent claim 1; and, under claim differentiation, Microsoft could not use them. Also, these descriptions apply to "arbitrary objects," not to an "arbitrary object framework," as these particular claims (13 and 14) confirm.

H. The Definition of "that separates a content of said computer application, a form of said computer application, and a functionality of said computer application"

This long phrase also appears in the preamble of the independent claims; and, as discussed above, it modifies or describes "arbitrary object framework." It appears in the preamble of claim 1, as well as in the preamble of claim 26, with the exception that in claim 26 "web site" replaces "computer application" in the three occurrences of "computer application." This phrase essentially provides that in practicing the method of the claims, one of the results that one can achieve is the separation of content, form, and functionality.

The first issue for the Court is whether this phrase (like the "arbitrary object framework" phase discussed above) is even a limitation of the claims that requires construction. Vertical maintains that it is not. Although the prosecution history, in a number of places, states that the invention of the claims of the '744 patent allows separation of form, content and function while the prior art does not, those statements highlight the results and benefits obtained by the present method. The body of the claims includes the creation of arbitrary objects which allows the separation, and, thus, the preamble does not add anything which the body does not include.

Should the Court decide that this phrase ("that separates…") is a limitation of the claims, then the parties propose the following definitions:

Vertical	Microsoft
Treating content, form and functionality as arbitrary objects.	In which the content, form and functionality of the computer application [web site] are entities that are independent of each other such that they do not reference each other, and no change to one requires a corresponding change in either of the other two. Prosecution History Disavowal: no class includes two or more of a content, a form, and a functional method.

Vertical submits that the following two sections of the specification compel adoption of its definition:

> More specifically, the present invention provides a method for generating software applications in an arbitrary object framework. The method of the present invention separates content, form, and function of the computer application so that each may be accessed or modified independently. (Column 2, lines 9-14, Exhibit A.)
>
> * * *
>
> The present invention provides an important technical advantage in that content, form, and function are separated from each other in the generation of the software application. Therefore, changes in design or content do not require the intervention of a programmer. This advantage decreases the time needed to change various aspects of the software application. (Column 2, lines 19-25, Exhibit A.)

The prosecution history also includes these statements in highlighting the benefits of the invention. Using arbitrary objects allows the independence and separation that is the central benefit of this invention. Therefore, the only meaning can be Vertical's construction.

The invention of the '744 patent includes creating and using arbitrary objects which enables the separation of form, content and function. But, the invention does not compel this separation every time. Neither the '744 patent nor its prosecution history requires that

the method perform this separation all the time. In fact, quite the contrary, the large number of descriptions for arbitrary objects make certain that an arbitrary object can contain these three components separately or in any combination. Thus, Microsoft's definition and its "Disavowal" statement do not find any support in the internal record or elsewhere.

VI. CONCLUSION

For the foregoing reasons, this Court should adopt Vertical's proposed constructions of the disputed terms and phases and eliminate the preamble terms from consideration. Should the Court decide that the preamble terms are claim limitations, then Vertical respectfully requests that the court adopt its definitions for those terms.

VERTICAL COMPUTER SYSTEMS, INC.

Eric M. Albritton
Texas State Bar No. 00790215
ALBRITTON LAW FIRM
P.O. Box 2649
Longview, Texas 75606
Tel.: (903) 757-8449
Fax: (903) 758-7397
ema@emafirm.com

Thomas John Ward, Jr.
Texas Bar No. 00794818
Ward & Smith Law Firm
P O Box 1231
Longview, TX 75606-1231
Telephone: (903) 757-6400
Facsimile: (903) 757-2323

Raymond P. Niro
Vasilios D. Dossas
Sally Wiggins
NIRO, SCAVONE, HALLER & NIRO
181 West Madison Street, Suite 4600
Chicago, Illinois 60602
(312) 236-0733
Fax: (312) 236-3137

Attorneys for Plaintiff

CERTIFICATE OF SERVICE

The undersigned certifies that the foregoing document was filed electronically in compliance with Local Rule CV-5(a). As such, this motion was served on all counsel who are deemed to have consented to electronic service. Local Rule CV-5(a)(3)(A). Pursuant to Fed. R. Civ. P. 5(d) and Local Rule CV-5(d) and (e), all other counsel of record not deemed to have consented to electronic service were served with a true and correct copy of the foregoing by email and/or fax, on this the 16th day of May, 2008.

Eric M. Albritton